LibCT
One Lib to Rule Them All

Andrey Vagin
Odin, Software Engineer

ContainerCon, 2015
Agenda

• History
• Namespaces and CGroups
• LibCT
• And Libcontainer
• Current state and future plans
History of Linux Containers

• 2002 Virtuozzo
• 2005 OpenVZ
• Linux-VServer
• 2006 Namespaces and Cgroups
• 2008 LXC (Linux Containers)
• 2010 Application Containers ...
History of Application Containers

- Systemd-nspawn
  - *Spawn a namespace container for debugging, testing and building*

- Docker
  - LXC
  - Libcontainer

- Rocket
  - systemd-nspawn
Namespaces

- **Mount (2.4.19 2002)**
  - Mount points
- **Network**
  - Network devices, stacks, ports, etc.
- **PID**
  - Processes
- **IPC**
  - System V IPC, POSIX message queues
- **UTS**
  - Hostname and NIS domain name
- **User (3.8 - 2013)**
  - Security-related identifiers and attributes
Other features of namespaces

- **User namespaces** allow to use namespace for *unprivileged* users
- Namespaces can be created *once* and used *many times*
- **Fast entering** into existing namespaces
- Ability to **isolate tasks** in context of one user
Usage scenarios

• Mount namespace
  - to construct a new root which contains only accessible paths (security)
  - Multitenancy

• Network namespace
  - to grant full access to network devices (Network Function Virtualization, Virtual Private Networks, etc)
  - to build complex network topology
  - to use iptables, traffic shaping, etc

• PID namespaces
  - to isolate independent group of processes (security)
Control Groups

- Cpu, cpuset, cpuacct
- Memory, hugetlb
- Blkio
- Devices
- net_cls, net_prio
- Freezer
- perf_event
The LibCT library

- Allows to build containerized applications
  - configure namespaces and cgroups
  - unprivileged containers
- In C and binding for other languages
- Cross-platform
Reasons for creating LibCT

• Complexity of low-level API

• Support of all kinds of containers
  - Linux Containers
  - OpenVZ
  - Solaris Zones
  - BSD jails

• Hide low-level API changes
Libcontainer and LibCT

- In Go / In C
- Both support back-ends
- Only Go / binding for other languages
- Easy for developing / works faster
- No fork() / ...

Libcontainer → runc
LibCT API

Session
- create()
- load()

Container
- namespaces
- cgroups

ProcessDesc
- Uid, Git, Groups

Process
- kill(), wait()
Example

```c
s = libct_session_open_local();

/* configure container */
ct = libct_container_create(s, "test");
libct_container_set_nsmask(ct,
   CLONE_NEWPID | CLONE_NEWUSER | CLONE_NEWNS));
libct_usersns_add_uid_map(ct, 0, getuid(), 1);
libct_usersns_add_gid_map(ct, 0, getgid(), 1);
libct_fs_add_mount(ct, "tmpfs", "/tmp", 0, "tmpfs", NULL))

/* configure process descriptor */
p = libct_process_desc_create(s);
libct_process_desc_set_caps(p, 0, CAPS_ALL);

/* Executing process*/
pr = libct_container_spawn_cb(ct, p, set_ct_alive, ct_alive);
libct_process_wait(pr, &status);
libct_container_destroy(ct);
libct_session_close(s);
```
Future plans

- Integration with applications
- Notifications
- Task-less containers
- Checkpoint/Restore (CRIU)
Thank You!

Andrey Vagin <avagin@openvz.org>
https://github.com/avagin/libct